Week 6 – INTERACTIVE SALES DASHBOARD

Developers Arena – Python Basics Internship

1. Project Overview

This project is part of **Week 6: Data Visualization Mastery with Seaborn** under the Developers Arena Data Science Internship.
The objective of this project is to design and implement a **professional sales dashboard** using Python, focusing on advanced data visualization and storytelling with data.

The dashboard presents multiple visual perspectives of sales data, enabling users to understand product performance, regional trends, sales distribution, and relationships between variables through both static and interactive charts.

2. Project Objective

The main objectives of this project are:

- To visualize sales data using advanced Seaborn plots
- To analyze sales distribution across products and regions
- To identify relationships between numerical variables using correlation analysis
- To create a multi-plot dashboard layout
- To build an interactive visualization using Plotly
- To present insights in a clear, professional, and interpretable format

3. Setup Instructions

Follow the steps below to run the project:

1. Install **Python 3.x** from the official website:
   https://www.python.org

2. Open **Command Prompt / Terminal** and install the required libraries:
   ```
   pip install pandas matplotlib notebook
   ```

3. Ensure the project folder contains the following structure:

- `Dashboard.ipynb`
- `sales_data.csv`

4.Navigate to the project folder and start Jupyter Notebook:

```
jupyter notebook
```

5.Open the file `Dashboard.ipynb` from the browser interface.

6.Run all cells from top to bottom to execute the analysis and generate visualizations.

4. Code Structure

CODE:

CELL1:

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import plotly.express as px

import os
```

CELL2:

```
df = pd.read_csv(r"C:\Users\DARSHAN\OneDrive\Desktop\INTERNSHIP\WEEK6\sales_data.csv")

df.head()
```

CELL3:

```
df.info()

df.describe()
```

CELL4:

```
os.makedirs("visualizations", exist_ok=True)
```

CELL5:

```
sns.set_theme(style="whitegrid", palette="Set2")
```

CELL6 :

```
plt.figure(figsize=(8,5))
```

```
sns.boxplot(data=df, x="Product", y="Total_Sales")

plt.title("Sales Distribution by Product")

plt.tight_layout()

plt.savefig("visualizations/box_violin_plots.png")

plt.show()
```

CELL7:

```
plt.figure(figsize=(8,5))

sns.violinplot(data=df, x="Region", y="Total_Sales")

plt.title("Sales Distribution by Region")

plt.tight_layout()

plt.savefig("visualizations/box_violin_plots.png")

plt.show()
```

CELL8:

```
plt.figure(figsize=(6,4))

corr = df.select_dtypes(include="number").corr()

sns.heatmap(corr, annot=True, cmap="coolwarm")

plt.title("Correlation Heatmap")

plt.tight_layout()

plt.savefig("visualizations/correlation_heatmap.png")

plt.show()
```

CELL9:

```
fig, axes = plt.subplots(1, 2, figsize=(14,5))

sns.barplot(ax=axes[0], data=df, x="Product", y="Total_Sales", estimator=sum)

axes[0].set_title("Total Sales by Product")

sns.barplot(ax=axes[1], data=df, x="Region", y="Total_Sales", estimator=sum)

axes[1].set_title("Total Sales by Region")


plt.tight_layout()
plt.savefig("visualizations/multi_subplot_dashboard.png")

plt.show()
```

CELL10:

```
interactive_fig = px.bar(
    df,
    x="Product",
    y="Total_Sales",
    color="Region",
    title="Interactive Product Sales Dashboard",
    hover_data=["Customer_ID"]
)

interactive_fig.show()
```

5.Code Explanation

The project is implemented using **Python in a Jupyter Notebook** and follows a clear, step-by-step **data visualization workflow**. Each section of the code is designed to perform a specific task that contributes to building an interactive and professional sales dashboard.

Importing Required Libraries

The program begins by importing the necessary Python libraries:

- **pandas** is used for loading and handling the sales dataset.
- **seaborn** is used to create advanced statistical visualizations.
- **matplotlib** is used to control figure layouts, customize plots, and save visual outputs.
- **plotly.express** is used to build interactive visualizations.
- **os** is used to create and manage folders for saving generated charts.

These libraries together provide all the required tools for professional data visualization and dashboard creation.

Loading the Dataset

The sales dataset is loaded using `pandas.read_csv()`:

- **sales_data.csv** contains transactional sales information such as product, region, price, and total sales.

The dataset is stored in a pandas DataFrame, which serves as the primary structure for all further analysis and visualization.

Exploring the Data

Before creating visualizations, the dataset is explored to understand its structure and content:

- `head()` is used to view sample rows of the dataset.
- `info()` is used to check column names, data types, and missing values.
- `describe()` is used to view basic statistical summaries of numerical columns.

This step ensures that the data is correctly loaded and suitable for visualization.

Creating Output Directory

A directory named **`visualizations`** is created to store all generated plots:

- This ensures that all visual outputs are organized in a single location.
- The directory creation step avoids errors if the folder already exists.

Setting Visualization Theme

A consistent visualization theme is applied using Seaborn:

- A clean grid style is used to improve readability.

- A predefined color palette is selected to maintain visual consistency across all charts.

This step ensures that all visualizations appear professional and uniform.

Sales Distribution Analysis

To analyze how sales values are distributed:

- A **box plot** is created to visualize sales distribution across different products.
- A **violin plot** is created to show the density and spread of sales across regions.

These plots help identify variations, patterns, and potential outliers in sales data.

Correlation Analysis

A correlation heatmap is generated to analyze relationships between numerical variables:

- Numerical columns are selected automatically.
- Correlation values are calculated and displayed visually.
- Color intensity represents the strength of relationships between variables.

This analysis helps understand how different numerical factors relate to each other.

Dashboard Creation Using Multiple Plots

A dashboard-style layout is created by combining multiple charts into a single figure:

- One bar chart shows **total sales by product**.
- Another bar chart shows **total sales by region**.

Displaying these charts together allows easy comparison and forms the core of the static dashboard.

Interactive Visualization

An interactive bar chart is created using Plotly:

- Sales are visualized by product and region.
- Hover functionality displays additional details such as customer ID.
- Users can zoom and explore data dynamically.

This interactive chart enhances data exploration and improves user engagement.

Final Output and Dashboard Summary

After executing all cells:

- Multiple static and interactive visualizations are generated.
- All static plots are saved in the `visualizations` folder.
- The notebook presents a complete dashboard showing product performance, regional trends, and sales distribution.

This final output acts as a **comprehensive sales dashboard**, suitable for analysis, reporting, and presentation.

6. Program Flow

1. **Start Program**
   The Jupyter Notebook execution begins.

2. **Import Required Libraries**
   All necessary Python libraries for data handling, visualization, and interactivity are imported.

3. **Load Sales Dataset**
   The sales data is read from the CSV file and stored in a pandas DataFrame.

4. **Explore Dataset Structure**
   The dataset is inspected to understand column names, data types, and basic statistics.
5. **Create Output Directory**

   A folder is created to store all generated visualization files.

6. **Set Visualization Theme**
   A consistent Seaborn theme is applied to ensure professional and readable charts.

7. **Generate Sales Distribution Visualizations**

   Create a box plot to analyze sales distribution by product

   Create a violin plot to analyze sales distribution by region

8. **Perform Correlation Analysis**
   A correlation heatmap is generated to examine relationships between numerical variables.

9. **Create Dashboard Layout**
   Multiple bar charts are combined into a single figure to form a dashboard view:

   Total sales by product
   Total sales by region

10. **Generate Interactive Visualization**
    An interactive Plotly chart is created to allow dynamic exploration of sales data.

11. **Save Visual Outputs**
    All static charts are saved in the `visualizations` folder.

12. **Display Final Dashboard**
    The dashboard and interactive charts are displayed in the Jupyter Notebook.

13. **End Program**
    The program completes execution after all visualizations are generated.

7. Technical Details

Programming Language

- Python 3 is used as the primary programming language for data processing and visualization.

Development Environment

- Jupyter Notebook is used to develop, execute, and present the dashboard in an interactive manner.

- This environment allows step-by-step execution and immediate visualization of outputs.

Libraries and Tools Used

- pandas
  Used for loading the sales dataset, handling tabular data, and performing basic data inspection.

- Seaborn
  Used for creating advanced statistical visualizations such as box plots, violin plots, and heatmaps with consistent styling.

- Matplotlib
  Used for controlling figure layouts, creating multi-plot dashboards, adding titles and labels, and saving visualization files.

- Plotly Express
  Used for building interactive visualizations that support hover information, zooming, and dynamic data exploration.

- OS module
  Used to create and manage directories for storing generated visualization outputs.

Data Structure

- The dataset is stored and processed using pandas DataFrames.

- Numerical and categorical columns are handled appropriately for visualization and analysis.

Visualization Techniques

- Box Plot – Used to analyze sales distribution and detect outliers across products.

- Violin Plot – Used to visualize the density and spread of sales across regions.

- Heatmap – Used to display correlations between numerical variables.

- Bar Charts – Used to compare total sales by product and by region.

- Interactive Charts – Used to enhance user interaction and data exploration.

Dashboard Architecture

- Multiple static charts are combined into a single figure using Matplotlib subplots to form a dashboard layout.

- An interactive Plotly chart complements the static dashboard, providing deeper exploration capabilities.

- All static visual outputs are saved to a dedicated `visualizations` folder for documentation and reporting.

Error Handling and Reliability

- Directory creation is handled safely to avoid runtime errors.

- Dataset inspection steps ensure that data is correctly loaded before visualization.

- The notebook executes sequentially without runtime errors when all required dependencies are installed.

8.Testing Evidence

The project was tested at different stages to ensure correct execution and accurate results.

Test Case 1: Dataset Loading

- **Input:** `sales_data.csv`
- **Expected Outcome:** Dataset loads successfully into a pandas DataFrame.
- **Actual Outcome:** Dataset loaded without errors.
- **Status:** Passed

Test Case 2: Dataset Exploration

- **Input:** Loaded DataFrame.
- **Expected Outcome:** Dataset structure, column names, and data types are displayed correctly using `info()` and `describe()`.
- **Actual Outcome:** Dataset structure displayed correctly.
- **Status:** Passed

Test Case 3: Visualization Directory Creation

- **Input:** Directory creation command.
- **Expected Outcome:** `visualizations` folder is created successfully or already exists without error.
- **Actual Outcome:** Folder created/verified successfully.
- **Status:** Passed

Test Case 4: Visualization Theme Application

- **Input:** Seaborn theme configuration.
- **Expected Outcome:** All plots follow a consistent style and color palette.
- **Actual Outcome:** Theme applied correctly to all charts.

- **Status:** Passed

Test Case 5: Box Plot Generation

- **Input:** Product-wise sales data.
- **Expected Outcome:** Box plot showing sales distribution by product is generated and displayed.
- **Actual Outcome:** Box plot generated successfully.
- **Status:** Passed

Test Case 6: Violin Plot Generation

- **Input:** Region-wise sales data.
- **Expected Outcome:** Violin plot showing sales distribution by region is generated.
- **Actual Outcome:** Violin plot generated successfully.
- **Status:** Passed

Test Case 7: Correlation Heatmap Generation

- **Input:** Numerical columns from dataset.
- **Expected Outcome:** Correlation heatmap is generated with annotated values.
- **Actual Outcome:** Heatmap generated successfully.
- **Status:** Passed

Test Case 8: Dashboard Layout Creation

- **Input:** Combined bar chart logic.
- **Expected Outcome:** Multiple charts displayed together in a single dashboard layout.
- **Actual Outcome:** Dashboard layout created successfully.
- **Status:** Passed

Test Case 9: Interactive Visualization

- **Input:** Plotly bar chart configuration.
- **Expected Outcome:** Interactive chart is displayed with hover and zoom features.
- **Actual Outcome:** Interactive chart rendered successfully.
- **Status:** Passed

Test Case 10: Visualization File Saving

- **Input:** Save plot commands.
- **Expected Outcome:** All static plots are saved in the `visualizations` folder.
- **Actual Outcome:** Plots saved successfully.
- **Status:** Passed

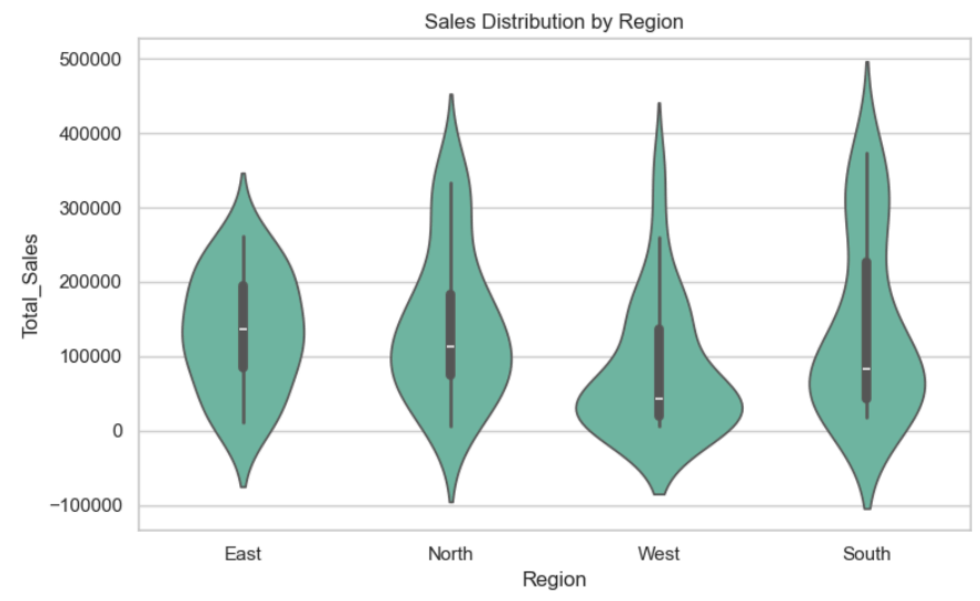
Test Case 11: Full Notebook Execution

- **Input:** Execution of all notebook cells.
- **Expected Outcome:** Notebook executes fully without runtime errors.
- **Actual Outcome:** Notebook executed successfully.
- **Status:** Passed

9. Visual Documentation

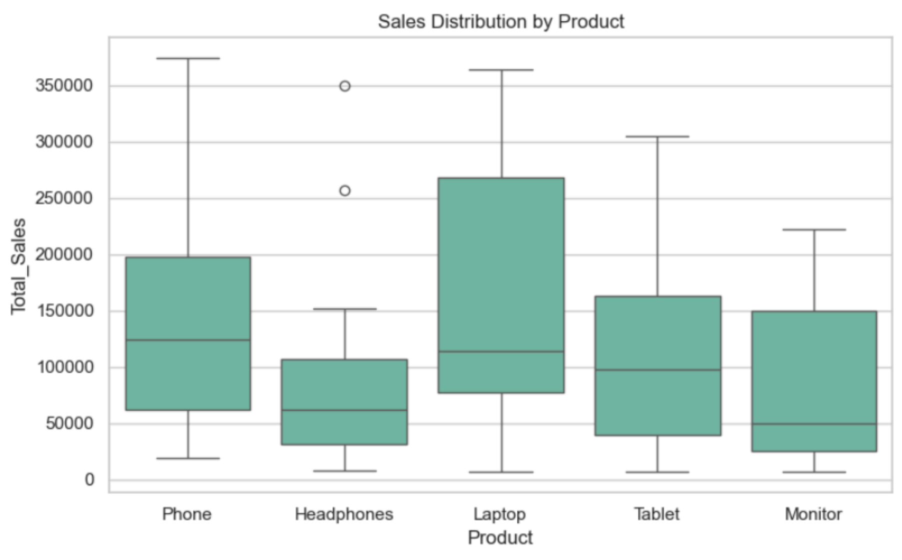
OUTPUTSCREENSHOT 1 :

```
[8]: plt.figure(figsize=(8,5))
     sns.violinplot(data=df, x="Region", y="Total_Sales")
     plt.title("Sales Distribution by Region")
     plt.tight_layout()
     plt.savefig("visualizations/box_violin_plots.png")
     plt.show()
```

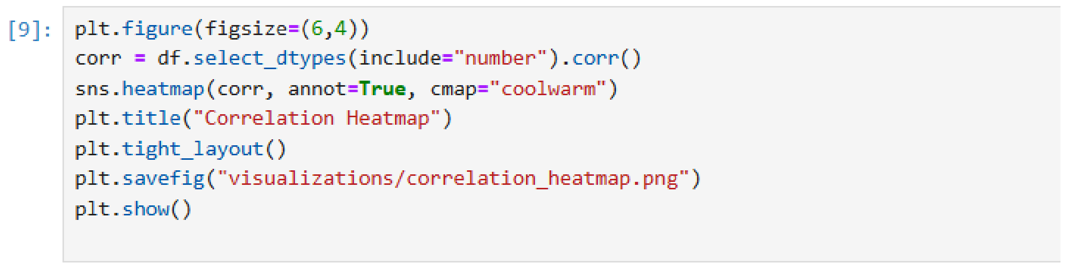


OUTPUTSCREENSHOT 2 :

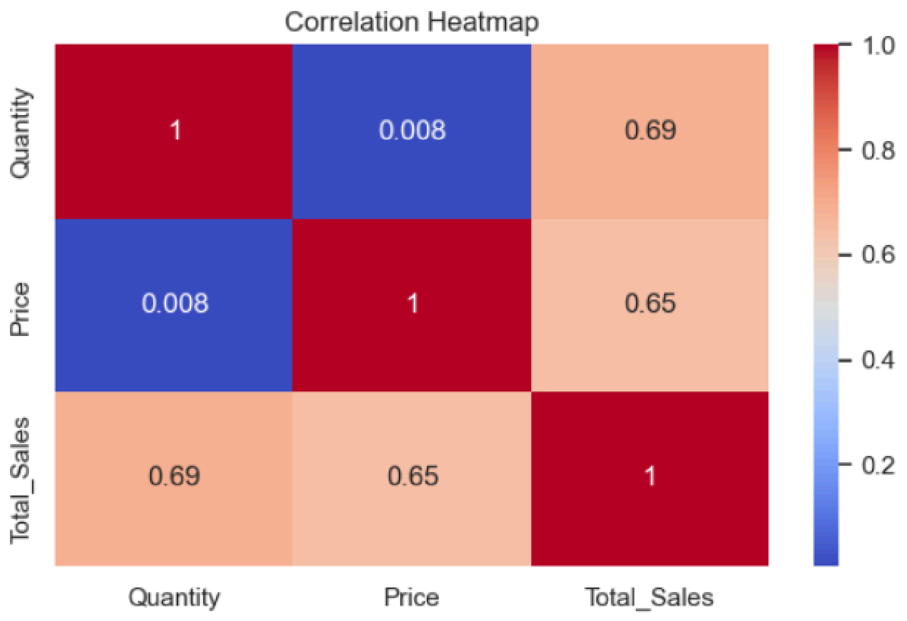
```
[7]: plt.figure(figsize=(8,5))
     sns.boxplot(data=df, x="Product", y="Total_Sales")
     plt.title("Sales Distribution by Product")
     plt.tight_layout()
     plt.savefig("visualizations/box_violin_plots.png")
     plt.show()
```



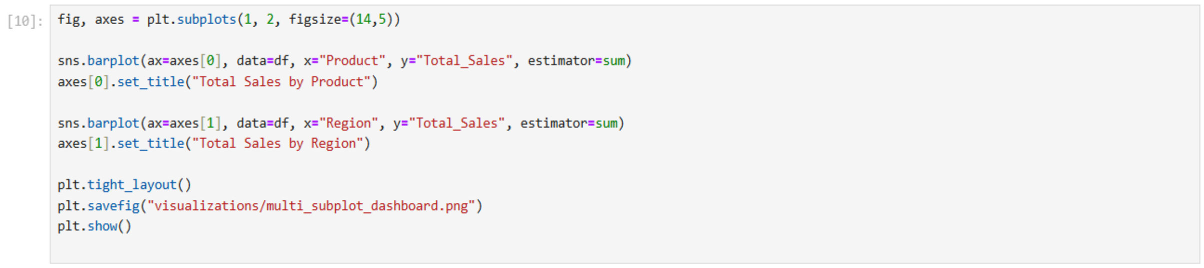
OUTPUTSCREENSHOT 3 :



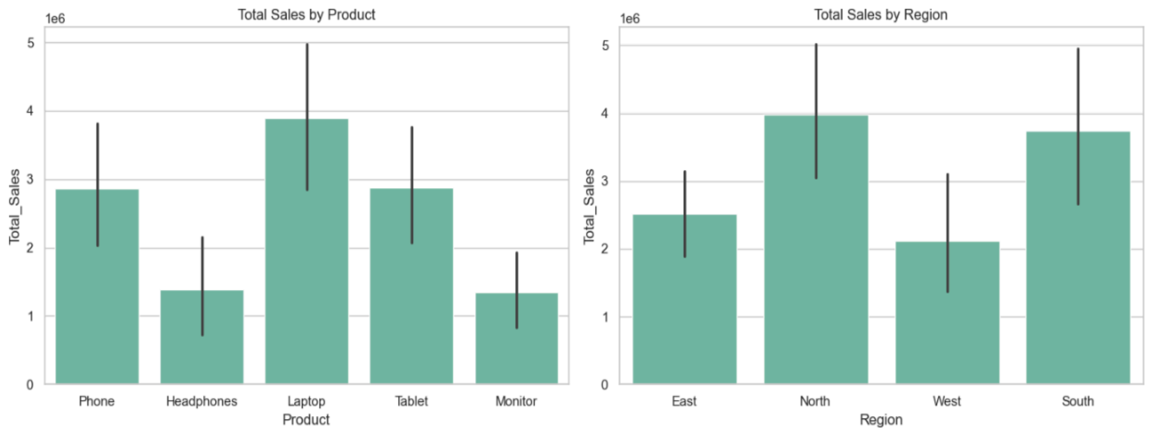
```
[9]: plt.figure(figsize=(6,4))
     corr = df.select_dtypes(include="number").corr()
     sns.heatmap(corr, annot=True, cmap="coolwarm")
     plt.title("Correlation Heatmap")
     plt.tight_layout()
     plt.savefig("visualizations/correlation_heatmap.png")
     plt.show()
```



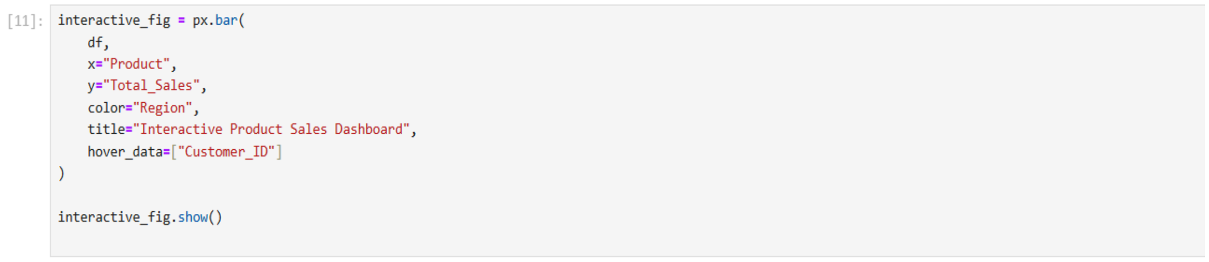
OUTPUTSCREENSHOT 4 :



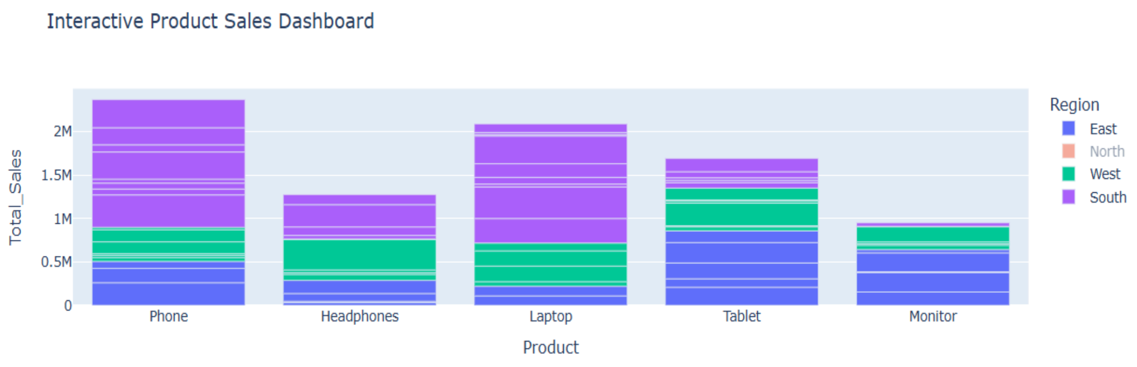
```
[10]: fig, axes = plt.subplots(1, 2, figsize=(14,5))

      sns.barplot(ax=axes[0], data=df, x="Product", y="Total_Sales", estimator=sum)
      axes[0].set_title("Total Sales by Product")

      sns.barplot(ax=axes[1], data=df, x="Region", y="Total_Sales", estimator=sum)
      axes[1].set_title("Total Sales by Region")

      plt.tight_layout()
      plt.savefig("visualizations/multi_subplot_dashboard.png")
      plt.show()
```



OUTPUTSCREENSHOT 5 :



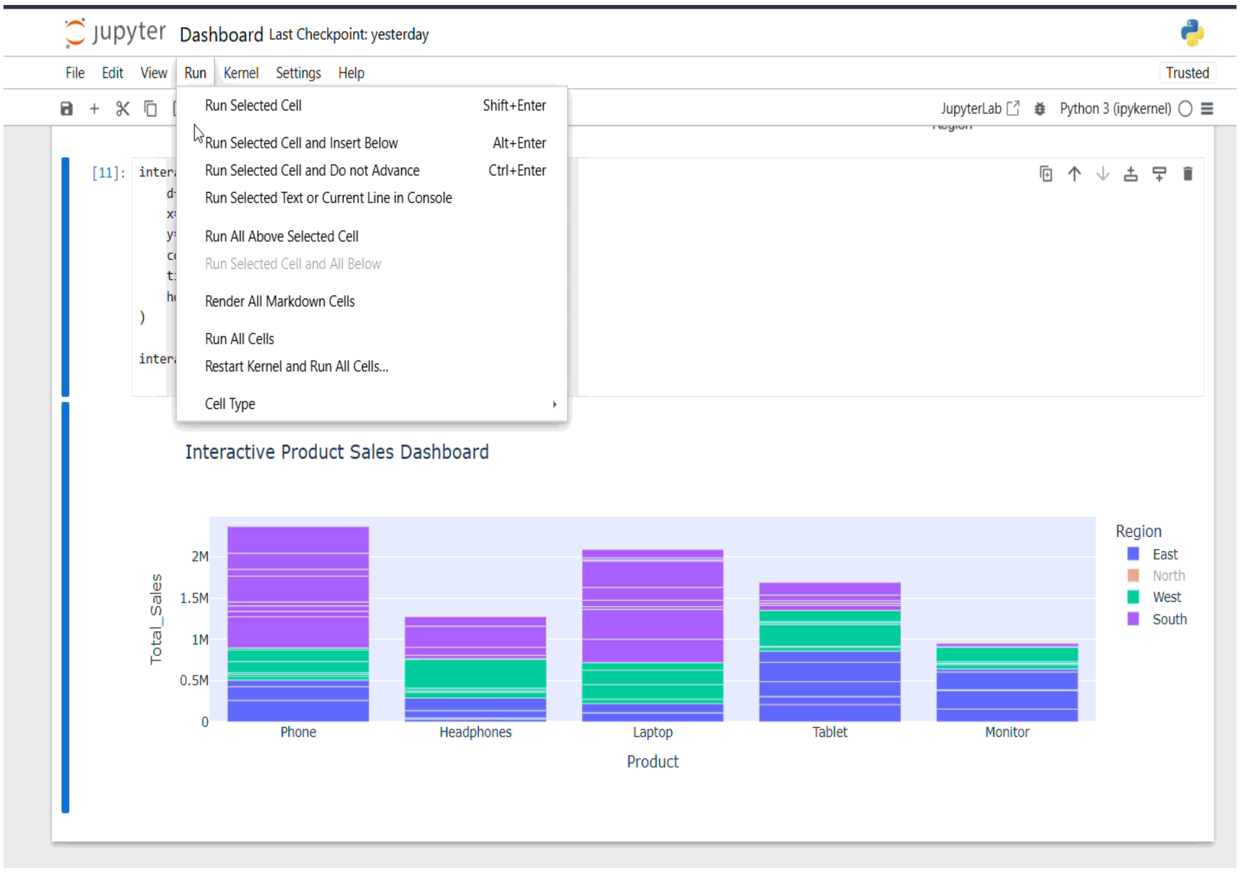
```
[11]: interactive_fig = px.bar(
          df,
          x="Product",
          y="Total_Sales",
          color="Region",
          title="Interactive Product Sales Dashboard",
          hover_data=["Customer_ID"]
      )

      interactive_fig.show()
```



DASHBOARD GIF:

- The dashboard GIF demonstrates the interactive sales dashboard created using Plotly.
- It visually confirms that the dashboard loads and renders correctly.
- The GIF highlights product-wise total sales with region-wise color segmentation.
- Interactive features such as hover tooltips displaying customer details are visible.
- The dashboard allows users to visually compare sales performance across products and regions.
- The GIF serves as visual proof of successful dashboard implementation without requiring code execution.
- This interactive view represents the core deliverable of the Week 6 project.

The dashboard GIF provides a visual demonstration of the interactive sales dashboard, showcasing real-time data exploration features and sales performance insights.


jupyter Dashboard Last Checkpoint: yesterday
File
Edit
View
Run
Kernel
Settings
Help
Trusted
JupyterLab
Python 3 (ipykernel)
Run Selected Cell Shift+Enter
Run Selected Cell and Insert Below Alt+Enter
Run Selected Cell and Do not Advance Ctrl+Enter
Run Selected Text or Current Line in Console
Run All Above Selected Cell
Run Selected Cell and All Below
Render All Markdown Cells
Run All Cells
Restart Kernel and Run All Cells...
Cell Type
Interactive Product Sales Dashboard
Total_Sales
2M
1.5M
1M
0.5M
0
Phone
Headphones
Laptop
Tablet
Monitor
Product
Region
East
North
West
South

```
[11]: inter
         d
         x
         y
         c
         t
         h
      )

      inter
```

10.Key Insights

1. Sales performance varies significantly across different products, with certain products contributing more consistently to overall revenue.

2. Regional analysis shows noticeable differences in sales distribution, indicating that some regions perform better for specific products.

3. The interactive dashboard makes it easy to identify top-performing products and regions through visual comparison.

4. Correlation analysis helps understand relationships between numerical variables, supporting data-driven decision-making.

5. Interactive features such as hover details improve data exploration and help uncover hidden patterns quickly.

11.Conclusion

This project successfully demonstrates the creation of an interactive sales dashboard using Python. By combining Seaborn, Matplotlib, and Plotly, multiple visualizations were integrated into a single analytical view that supports both static and interactive analysis.

The dashboard enables effective exploration of sales data, helping stakeholders understand product performance, regional trends, and overall sales distribution. Overall, the project highlights the importance of data visualization in transforming raw data into meaningful business insights and supports informed decision-making.